UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/815,018	03/26/2004	Gregor K. Frey	6570P033	1010

8791 7590 03/17/2009

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
03/17/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/815,018	FREY ET AL.	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *23 February 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-16,18-20,22-25,27 and 28* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-16, 18-20, 22-25, 27-28* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *3/2/09; 11/20/08*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 2/03/09.

As indicated in Applicant's response, claims 1, 10, 16, 18-20, 22-23, 27 have been amended, and claims 17, 21, 26 canceled. Claims 1-16, 18-20, 22-25, 27-28 are pending in the office action.

Claim Objections

2. Claim 22 is objected to because of the following informalities: there is a missing "," between "for the formatter" and "cause the apparatus to provide". Appropriate correction is required.

3. Claims 1, 16 are objected because: a) the "," after "including" (memory, the computer system further including," – cl. 1, line 4, cl 16 line 3) should have been a ":" and b) a properly *conjugated* verb form is deemed missing in the clause "wherein the formatter … to format the output from one of the tracing module ...' (cl. 1, line 19; cl. 16, li. 24)

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-16, 18-20, 22-25, 27-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over James **Hart**, "Early Adopter: J2SE 1.4", chapter 5, September 2001, Wrox Press, pp. 1-12 (hereinafter Hart – refer to 'EarlyAdopter_DS.pdf' in PTO -892) in view of **APA** (Admitted Prior Art: see Specifications pg. 4, para 0002-0009) and further in view of Sun

Microsystems, J2SE: "Package Java.util.prefs" pp. 1-2 (**Prefs_1**) and "Class Preferences" , pp. 1-24 (**Prefs_2**), Copyright 2003 - < http://java.sun.com/j2se/1.4.2/docs/api/java/util/prefs/package-summary.html>

As per claim 1, Hart discloses an integrated tracing and logging system employed within a network (e.g. *logging server environment, client desktop* – pg. 2 top para) comprising a computer system having a processor coupled with a memory, the computer system further including:

a tracing module (e.g. *methods ... can be used ... debugging trace of program activity* – pg. 6, bottom half; *part of this API ... throws IllegalArgumentException* – pg. 7, top half) associated with specified program code of an application, the tracing module to receive via an application programming interface (API) and process tracing method calls generated by the application (the Logging API, pg. 2; void Methods – pg. 5 reads on via an API);

a logging module associated with specified categories related to the network (e.g. Hart: pg. 2 top para ; pg. 4, top – Note: complex application with subsystems or subclasses reads on specific categories of complex network application), the logging module to receive via the API and process logging method calls (e.g. Logging Methods - pg. 5-7) from network components associated with the categories (*distributed applications... number of networked machines*, pg. 1; pg. 2 top para); and

a formatter coupled to the tracing module and the logging module, the formatter to receive an output from one of the tracing module and the logging module (formatters – 2nd para , pg. 5; pg. 2 bottom Figure),

wherein the formatter operates as to format the output from the one of the tracing module and the logging module (e.g. string message … formatters – pg. 5, 2nd para ; what order … what parameters – bottom half, pg. 6);

and an output destination (e.g. *different destination; log files, the console, ... in-memory buffer* – pg. 2, top para; *fired off ... different types of storage* – pg. 1) to receive the formatted output (*Logging API*... **Loggers**: *pass messages for logging* – pg. 2; *Messages are passed* - pg. 2 bottom)of the at least one of the tracing module and the logging module.

Hart does not explicitly disclose *tracing module associated with specified program code regions of an application* to receive and process tracing method calls generated by the application *when the specified program code regions are executed*. APA teaches J2SE loggers organized in *namespaces* and tightly used with log messages in Java applications (Specifications para 0003 - pg 2)and tracing being tightly used by developers using logging techniques (Specs para 0007-0008 -pg. 4) in terms that tracing tool being equally used with logging also require sending messages to console or other output destination. Analogous to APA in terms of user level for setup loggers including passing of messages in application tracing and passing of logging methods calls (see Hart: pg. 3), Hart teaches J2SE and naming convention for each logger with a *setLevel* based on a particular *namespace* (pg. 4 bottom) with the hierarchy of logger packages and utilization within complex applications of loggers which can be instantiated as needed for subclasses of the applications. According to which, distinct error logs or levels settings (Hart: pg. 4, top) are in such number that a manager class could be instantiated to alleviate issues from the proliferated number of loggers. Based on user-controlled levels setting and subclasses of loggers instantiating as set forth above, it would have been obvious for one

skill in the art at the time the invention was made to associate the debug and tracing module as set forth above by Hart in handling complex large user applications, so that tracing calls invoked from Hart's user's debug/tracing scheme (regarding user specified applications) are directed for specific regions or locations, so that level setting can be applied for each such regions and for which a specific logger can be communicated with messages based on the instantiated namespace allocated for the instance of logger and the level for said region among many more subdivisions of a complex application.

Hart does not explicitly disclose formatter including *a configuration file storing a format definition for the formatter; the configuration file of the formatter further to receive a change to the format definition for the formatter during a runtime of the integrated tracing and logging system, wherein receiving the change to the format definition for the formatter does not require recompiling of any source code* of the integrated tracing and logging system.

Hart discloses a tracing and logging working in conjunction with a manager object in terms that a file preferences is read (Hart: *this file is read* - pg. 7 bottom) by the manager based which the logging classes are subsequently instantiated, in terms that the file contains handlers, default settings therefor, and default level as this in set to INFO, all of such setting taught in *J2Se/1.4/docs/util* site (pg. 7 bottom, pg. 8 top). In fact, J2SE's **Prefs_1** discloses a Preference class to collect preference data (see Prefs_1: pg. 1) with formatting type methods for this class to represent *string or ByteArray, Default keys* (Prefs_2: pg. 15-16, 19-20) and, with respect to reading a file, **Prefs_2** shows that this collection include nodes and path names represented in <> or "/" separated identifiers, wherein the Preferences object can receive method calls to modify node and path along with availability of all such data being backed by persistent file

storage like a *preferences* file exported as XML document with associated notation and corresponding Doctype (see Prefs_2: bottom, pg. 1 to pg. 2, 3). Based on the message communication between loggers and formatters (see Hart, Figure pg. 2) and listening capability of the Preferences object, it would have been obvious for one skill in the art at the time the invention was made to implement the level setting and message passing between formatters and different loggers in a complex application wherein tracing module passed logging methods to these loggers so that formatting is based a exportable Preferences document being referred to as set forth above; that is, reusing information from a configuration file that is consulted based on the implementation of Preference class or from runtime persisting of default configuration as above OR a exportable version of preference data set (via methods of a class) for formatting being backed as XML. One would be motivated to do so because formatting would benefits from dynamics of the Preference object changes, in terms that not only can this Preference class receives dynamic calls to have the hierarchy of name/path modified (i.e. *change to the format definition for the formatter during a runtime*)but also can have a backup persistent storage in terms of a XML document being exportable and used as consultation point for the Logging scheme to start as set forth above in Hart, without having to affect any source code or recompilation resources(i.e. *not require recompiling of any source code of the integrated tracing and logging system*), because since the Preferences data as backup XML file can be read as configuration startup and easily exported to other part of a NW complex system, the default preference and level setting for a plurality of loggers for parts of said complex application can be used to support tracing of related regions based on this available configuration file.

Nor does Hart disclose wherein the formatter operates as to format the output from the one of the tracing module and the logging module *according to the changed format definition*. But based on the Preference file operating as backup and Preference class with dynamic capability to receive calls to change hierarchy of names and path as set forth in Prefs_2 during asynchronous request from loggers and tracing method calls as set forth above in Hart's use of a management scheme, the formatting of messages via filters (see Hart Figure, bottom pg. 2) such that the formatting is based on the latest changes in format definition would have been obvious in view of the rationale as set forth above.

As per claim 2, Hart (in view of Prefs_2) discloses a markup language formatter (e.g. filters and formatters … information about logging event – pg. 5, 2nd para; *XML file, level threshold set to INFO ... precise configuration ... Java properties format* - pg. 7, bottom; bottom half pg. 8; see **Prefs_2**: XML document - pg. 2)

As per claims 3-4, and 6, Hart (in view of the persisted XML file in Prefs_2) discloses wherein one or more properties (e.g. log.severe – pg. 8, top; *more information ... it also contain an exception* – 2nd para, pg. 5) of the formatter are defined in the configuration file (see Prefs_2: pg. 19-20 – see XML persisted file in claim 1); wherein the configuration file (by way of obviousness in claim 1 – see **Prefs_2**: XML document - pg. 2) includes an identifier (e.g. *XML file, level threshold set to INFO ... precise configuration ... Java properties format* - pg. 7, bottom; *public class Logging* – pg. 8; *<class>Logging</class>* - pg. 8, bottom; *LogRecord's parameters* – bottom pg. 10) to identify the formatter (Note: logging class with Java handlers using INFO set from the XML reads on formatter); wherein the configuration file defines the

message format for the received message, the message format including one or more fields (e.g. *XML details ... <record> ... </record>* pg. 8-9).

As per claim 5, Hart (in view of Prefs_2) discloses XML persisting of collection of preferences node as a file based on listener to change node data of the Preferences class (see claim 1) hence markup language with tag identifier and value of tag (see Hart: <date> … </date>, <logger> … </logger> pg. 8; *998524070390, com.wrox.ea.j2se.utilities.Logging* – pg. 8) would have rendered the one or more properties are formatted as key-value-pair properties, each key-value pair having a key to specify an attribute and a value to provide a definition for the specified attribute obvious in view of the teaching in Prefs_2.

As per claim 7, Hart discloses (based on the rationale of claim 6) wherein the one or more fields of the message format includes

at least one of a timestamp field (e.g. *date, millis* – pg. 8, bottom – Note: record storing date and millis from a XML used for firing message reads on time of received message) to indicate a time for the received message;

a location of origin field to indicate a source (String sourceClass – pg. 5; <method>, pg. 8) of the received message;

a thread identifier field to indicate a thread (<thread> - pg. 8, bottom) associated with the received message;

a message severity indicator field to indicate a severity level (Level level – pg. 5; warning – pg. 8 bottom) of the received message; and

a message identifier field to identify the received message (String msg – pg. 5; <message> pg. 8, bottom).

As per claims 8-9, Hart discloses wherein the output destination is at least one of a trace file; and a log file, a console (e.g. *different destination; log files, the console ... in-memory buffer* – pg. 2, top para).

As per claim 10, Hart discloses a computer-implemented method employed within a network comprising:

creating an instance of a tracing controller associated with specified program code of an application, the tracing controller instance to receive and process tracing method calls generated by the application (refer to claim 1);

creating an instance of a logging controller (Hart: pg. 2 top para ; pg. 4, top – Note: complex application with subsystems or subclasses and a manager logger reads on controller for specific categories of complex network application) associated with specified categories related to the network, the logging controller to receive and process logging method calls from network components associated with the categories;

providing a common application programming interface (*the Logging API*: … framework of classes pg. 2, top)of the tracing controller instance and the logging controller instance, whereby the tracing controller instance and the logging controller instance are accessed (API framework - pg. 7 bottom; *we can use the LogManager's ability to control the levels* – pg. 9 top);

creating an instance of a formatter coupled to the tracing module and the logging module (refer to claim 1),

receiving at the formatter an output from one of the tracing module and the logging module (refer to claim 1) and

formatting at the formatter the output from the one of the tracing module and the logging module, the formatting (refer to claim 1).

Hart does not explicitly disclose *tracing module associated with specified program code regions of an application* to receive and process tracing method calls generated by the application *when the specified program code regions are executed.* But this application has been addressed in claim 1.

Nor does Hart explicitly disclose the formatter including a configuration file storing a format definition for the formatter; after the creating the instance of the formatter, changing the format definition stored in the configuration file, wherein changing the format definition does not require a recompiling of any source code. The concept of dynamically changing the collection of data in the Preference file with listener in place (see Prefs_1, pg. 1; Prefs-2: pg. 22) discloses that runtime listening of the preference object for changes to setting to user's preferences after the logger and formatter are already executing, such that by presenting a configuration made exportable based on the dynamic recording of Preferences data as in Prefs_2 (see claim 1), the effect of Hart formatting using settings of stored configuration file after the formatter instance is started would be evident. The rationale regarding the configuration file used in formatting in conjunction with message communication with the tracing/logging controller has been set forth in claim 1, and herein applied.

Nor does Hart disclose formatting at the formatter the output from the one of the tracing module and the logging module, the formatting *according to the changed format definition of the configuration file.* This limitation has been rendered obvious in view of the above rationale.

As per claims 11-12, refer to claims 2, 4 (Note: selected formatter functions to configure message using the XML in claim 2 reads on configuring for the selected formatter).

As per claims 13-14, refer to claims 6-7.

As per claim 15, Hart discloses a filter to the specified output destination to selectively filter the message.

As per claim 16, Hart discloses system comprising a computer system having a processor coupled with a memory, the computer system further including means for:

creating an instance of a tracing controller associated with specified program code of an application (refer to claim 10), the tracing controller instance to receive via an application programming interlace (API) and (refer to claim 10) process tracing method calls generated by the application;

creating an instance of a logging controller associated with specified categories related to the network, the logging controller (refer to claim 10) to receive via the API and process logging method calls from network components associated with the categories (refer to claim 1),

creating an instance of a formatter to receive an output from one of the tracing controller instance and the logging controller instances (refer to claim 10), wherein the formatter instance further to format the output from the one of the tracing controller instance and the logging controller instance (refer to claim 10), and

an output destination to receive the formatted output of the one of the tracing controller instance and the logging controller instance (refer to claim 10).

Hart does not explicitly disclose *tracing module associated with specified program code regions of an application* to receive and process tracing method calls generated by the

application *when the specified program code regions are executed*. But this application has been addressed in claim 1.

Nor does Hart explicitly disclose formatter instance including a *configuration file* storing a *format definition for the formatter* instance; and changing the format definition stored in the configuration file for the formatter instance, wherein the *changing the format definition does not require a recompiling of any source code*. But the formatter's configuration file with format definition being changed without recompiling source code has been rendered obvious in claim 1.

Nor does Hart disclose formatting according to the changed format definition stored in the configuration file; but this have been rendered obvious based on the rationale regarding formatting and configuration file from above.

As per claims 18-19, refer to claims 13-14.

As per claim 20, Hart (in view of APA, Prefs_1, Prefs_2) discloses article of manufacture comprising an electronically accessible medium providing instructions that, when executed by an apparatus, cause the apparatus to:

create an instance of a tracing controller associated with specified program code regions of an application, the tracing controller instance to receive and process tracing method calls generated by the application when the specified program code regions are executed (refer to claim 10 and rationale of obviousness);

create an instance of a logging controller associated with specified categories related to the network, the logging controller to receive and process logging method calls from network components associated with the categories (refer to claim 10);

provide a common application programming interface of the tracing controller instance and the logging controller instance, whereby the tracing controller instance and the logging controller instance are accessed (refer to claim 10);

create an instance of a formatter coupled to the tracing module and the logging module, the formatter including a configuration file storing a format definition for the formatter: after the creating the instance of the formatter, change the format definition stored in the configuration file. wherein changing the format definition does not require a recompiling of any source code (refer to claim 10 and rationale of obviousness);

receive at the formatter an output from one of the tracing module and the logging module: and format at the formatter the output from the one of the tracing module and the logging module, the formatting according to the changed format definition of the configuration file (refer to claim 10 and rationale of obviousness).

As per claim 22, refer to claim 13.

As per claim 23, Hart (in view of APA, Prefs_1, Prefs_2) discloses an apparatus comprising an application; and a processor and logic executable thereon to:

create an instance of a tracing controller associated with specified program code regions of the application, the tracing controller instance to receive and process tracing method calls generated by the application when the specified program code regions are executed;

create an instance of a logging controller associated with specified categories related to a network, the logging controller to receive and process logging method calls from network components associated with the categories;

provide a common application programming interface of the tracing controller instance and the logging controller instance, whereby the tracing controller instance and the logging controller instance are accessed;

create an instance of a formatter coupled to the tracing module and the logging module, the formatter including a configuration file storing a format definition for the formatter;

after the creating the instance of the formatter, change the format definition stored in the configuration file, wherein changing the format definition does not require a recompiling of any source code;

receive at the formatter a message from one of the tracing module and the logging module: and format at the formatter the message from the one of the tracing module and the logging module, the formatting according to the changed format definition of the configuration file;

all of which have been addressed in claim 10.

As per claims 24-25, refer to claims 2-3.

As per claims 27-28, refer to claims 13-14.

Response to Arguments

6. Applicant's arguments filed 2/03/09 and bearing solely on the 'configuration file limitation' in a runtime 'formatter' with parameters changing context have been fully considered but they are moot in light of the new grounds of rejection which have been necessitated by the Amendments.

Conclusion

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.
Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

March 13, 2009